CS 458 A1-MILESTONE

Chia Ching Chuen |WATID: CCCHIA | STUDENT
NUMBER: 20755359

Abstract
Explaination of exploit 1 and exploit 2

Sploit1.c

This exploit program will overflow the option -s of the program pwgen. As the program code for the parameter -s, accepts a user's input for the insertion of the string, it seems like a good place to start trying to exploit the program. Firstly, after looking through the code of the pwgen, in the code, I find that there exists a line of code :

```
args = parse_args(argc argv);
```

Inside of parse_args function, specifically, the line:

strcpy(args.salt, optarg);

as strcpy doesn't check whether the string variable is big enough for the string to be copied over, we are able to use this for a buffer overflow attack to change the return address after the function ends.

For now, the location of the return address, the amount of space to overwrite with NOPs and the shellcode before inserting the address of the location to jump to and the address to jump to is needed. first, I have to know where args.salt is located at specifically, using gdb, we are able to find the location at 0xffbfd7df by using the "print &args.salt" command.

As we checked, the distance from salt to the return address is roughly 557, we construct the buffer with a size of 557 bytes.

We first constructing the buffer with NOP, the shellcode and the address of args.salt repeatedly, to jump to in order. This will make the RA hopefully jump to the NOP or the shellcode.

By now, after compiling the program, and running it, a segmentation fault will be shown where my return address is a jumbled up of 0xffbfd7df address. After wrapping the address around and compiling it, the vulnerability is exploited and a root shell is spawned.

Sploit2.c

In print_usage function, we can see that when printing the help menu, its first uses the argument 0, which will be whatever command that was used to initialise the program and after which it prints the buffer with printf. However, in the printf statement, it directly prints the buffer without using any format string. To exploit this, we will first need a payload which includes the shellcode, the address of the payload and the EIP to overwrite with %n. as the argument needed in this exploit is argument 0, we will need to be able use that. To use the filename as an attack vector, we will have to use a symbolic link and link a newly created link file to attack the pwgen. However, there is an even simpler way of doing it. As the execve takes in another set of argument as the proper argument instead of using the filename as the argv[0], we are able to insert the payload into argv[0], and print the shellcode, doing so, it will allow us to achieve a rooted shell.